6: Introduction to 586 Engine and PIO

1-Introduction

The objectives of this lab are:-

A- To get the students familiar with the SC-586 microprocessor developing board.
B- To teach the students how to program a general input/output (GPIO), and how to write/ read the digital signals.

2-Theory

586 Engine is a high performance 32-bit 586 CPU with Floating Point Unit (FPU), ADC, DAC, UARTs, with the following specifications:

- Measures 3.6x2.3x0.3 inches
- 32-bit CPU (AMD SC520), x86 compatible
- 100/133 MHz system clock
- High performance hardware floating point coprocessor
- Power consumption: 440 mA at 5V
- Temperature: -40 to +80 Celsius
- Up to 512 KB Flash/EPROM
- Up to 512 KB battery-backed SRAM
- 114 byte internal RAM
- 32 PIOs
- 7 timers
- 15 interrupts
- One synchronous serial-port, three UARTs
- 19 channels 12-bit ADC
- 8 channels 12-bit DAC
- Real-time lock, battery

The 586-Engine (TM) (5E) is a C/C++ programmable controller based on a 32-bit 100/133 MHz AMD Elan SC520. The SC520 integrates an Am586 CPU and a high performance ANSI/IEEE 754 compliant 64-bit hardware floating point unit (FPU). It is designed for embedded systems that require high performance, PC-compatibility, compactness, and high reliability at a low cost. Whether you use it as the processor core component of an OEM product, as part of a smart sensor, or as a node in a distributed microprocessor system, building your product around the 5E will reduce development time and costs, minimize technical risks, and allow you to deliver a more reliable end product.
The FPU provides arithmetic instructions to handle numeric data and transcendental functions for sine, tangent, logarithms, etc, making this controller useful for intensive computational applications. It is estimated to be 10-50 times faster than software-emulate on an 8/16-bit controller without a FPU.

The 586-Engine supports up to 15 external interrupts. There are a total of seven timers, including one programmable interval timer (PIT) that provides three 16-bit PIT timers and three 16-bit GP timers, plus a software timer. These timers can support timing or counting external events. The software timer provides a very efficient hardware time base with microsecond resolution. A real-time clock (RTC) provides time-of-day, 100-year calendar and 114 bytes of battery backed RAM.

Two industrial-standard 16550-compatible UARTs support baud rates up to 1.152 M baud. One synchronous serial interface (SSI) supports full-duplex bi-directional communication. One additional optional UART (SCC2691) can be installed.
The 586-Engine(TM) boots from on-board 256K 16-bit ACTF Flash, and supports up to 256K 16-bit battery-backed SRAM. SDRAM and DMA are not supported. Up to 1GB memory expansion can be added with MemCard-A and PCMCIA ATA Flash cards.

There are 32 programmable multifunctional I/O lines (PIO) that can be used as general I/O or other functions. Two supervisor chips monitor 5V and 3.3V and provide power failure detection, watchdog and system reset. The 2.5V is used for the SC520 core and 3.3V for the I/O operation. Signal lines on headers are 3.3V output, and 5V maximum input.
All components are soldered on board for highest reliability. The 586-Engine^(TM) can be powered with a single regulated 5V with the on-board 3.3V and 2.5V regulators. It also can be powered by external regulated 5V, 3.3V, and 2.5V.
A total of 19 analog inputs are supported with a serial ADC (TLC2543, 11 inputs, 20KHz sample rate, 12-bit, 0-REF+) and a parallel ADC (AD7852, 8 inputs, 100KHz sample rate, 12-bit, 0-5V). Two serial DACs (LTC1446, 2 channels, 0-4.095V) and a parallel 12-bit DAC (DA7625, 4 channels, 0-2.5V) can be in-stalled to provide a total of 8 analog outputs.
The 586-Engine shares a similar pin-out and physical dimensions with the i386-Engine, A-Engine and V25-Engine-LM. As compared to these other engine-based controllers, the 5E offers substantially more computing power at greater per-unit costs and power requirements.

The EZ-586 Tutor PIO pins are 3.3V output and all inputs are 5V tolerate. **Absolutely no voltage above the 5 volt should be applied to any pins. Over 5 volt can cause a permanent damage.** Pins P0-P1, P27, and P31 are free to use. A P10 line can be configured to operate as an output or an input with a weak internal pull-up resistor.

Paradigm C++ is a complete Integrated Development Environment (IDE) for developing real and protected mode x86 embedded system applications. The integrated source level debugger for all the Intel 186 and 386 microprocessors supports stand-alone debugging or can be used as a front-end for popular in-circuit emulators from Applied Microsystems and others. Paradigm C++ shares a common user interface with full support for source level debugging of embedded C/C++ applications. Featuring watches, inspectors, and a variety of memory, I/O and real-time operating system viewers, Paradigm C++ provides a high level of control and visibility of the target system. When necessary, the ability to drop down to a low level CPU or memory view is always available. Paradigm C++ includes the Paradigm PDREMOTE/ROM debugging kernel, ready to be customized for your unique target system. Customized versions of Paradigm C++ add support for hardware breakpoints, real-time trace, overlay memory, and other custom hardware found in popular in-circuit emulators.

3. Equipment Used

- **EZ-586 Tutor** from AMS that contains a 586 (5E) controller from Tern. **(Note that PIO 6, 7, 8, and 9 are not working).**
- Paradigm C/C++ Software
- Digital training kit ETS-7000.
- Oscilloscope.
- Logic analyzer.

4. Procedures

1- Demonstrating the hardware and software.

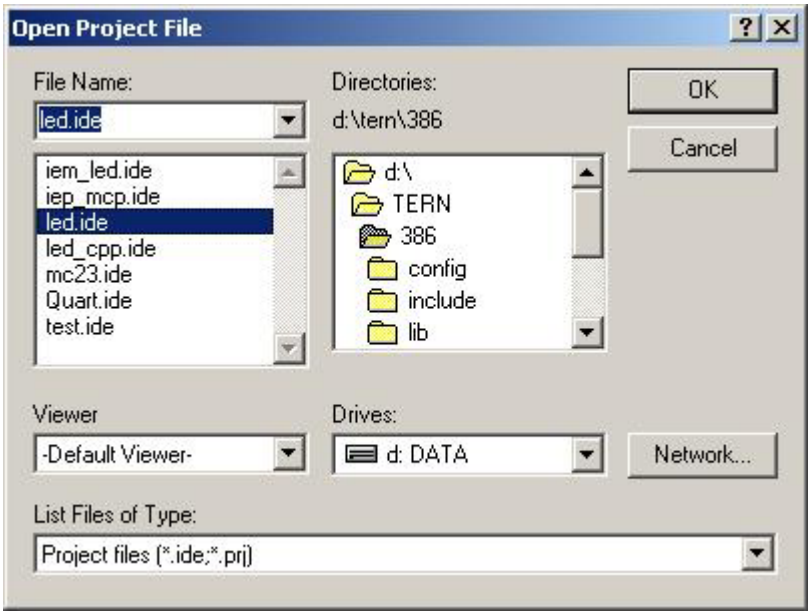
1.1 Connect the DB9 connector from COM1 or COM2 of the PC's ports to J3 on EZ-586 Tutor. LD3 turn on indicating that the selected PC's COM port is active.

1.2 Make sure SW1 (all switches are set to off position) on the EZ-586 Tutor.

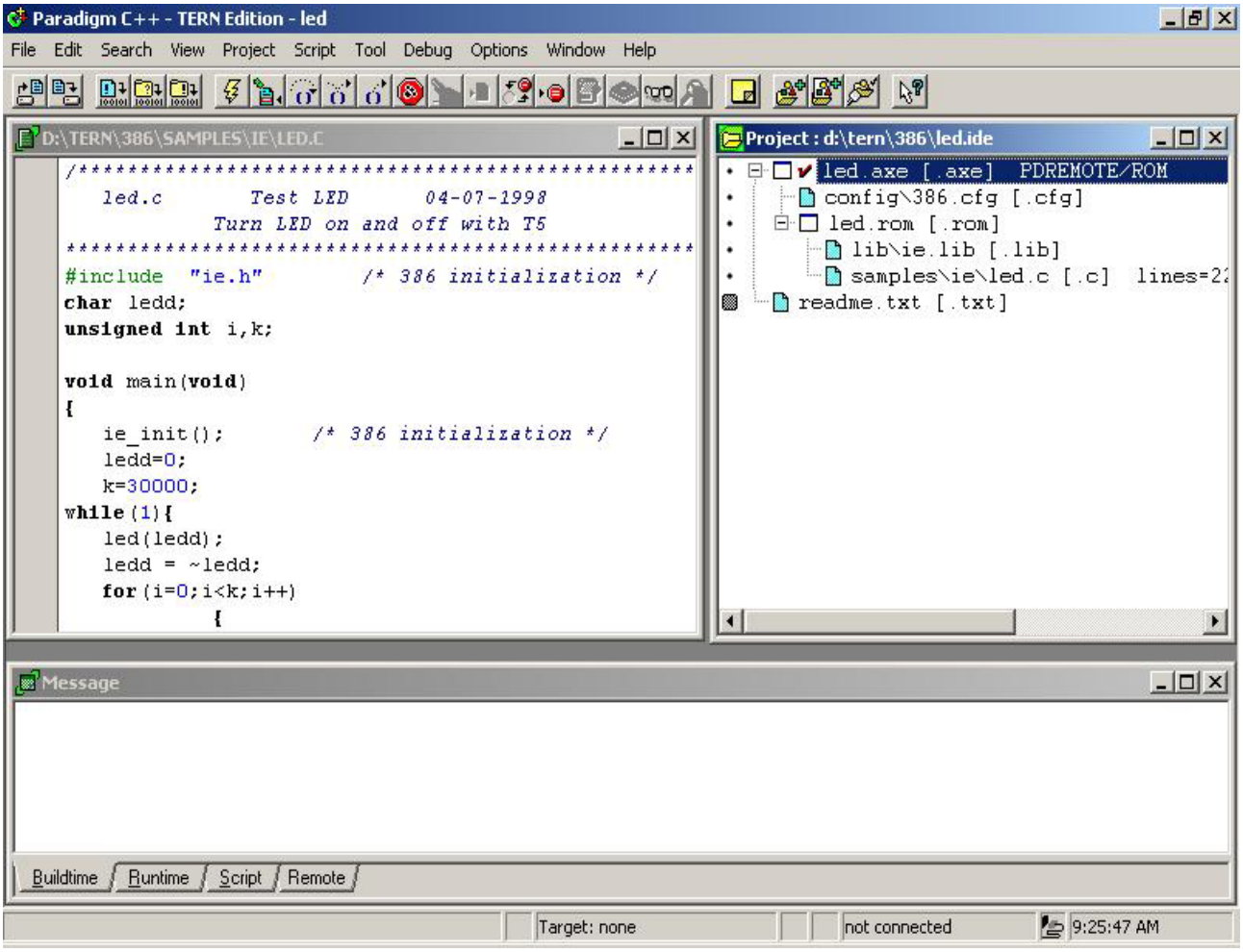
1.3 Make sure that the step 2 Jumper is connected to J1(pins 38-40) on the tutor in order to enable you to download the written program from the PC to the EZ tutor.

1.4 Connect the wall transformer (+9VDC) to J1 of the EZ-586 Tutor. LD1 turn on indicating that the board is powered up. LD2 and LD5 also turn on. Here look at L1 at the 586 engine it will blink twice then stays on, this indicate that the board is ready to download programs.

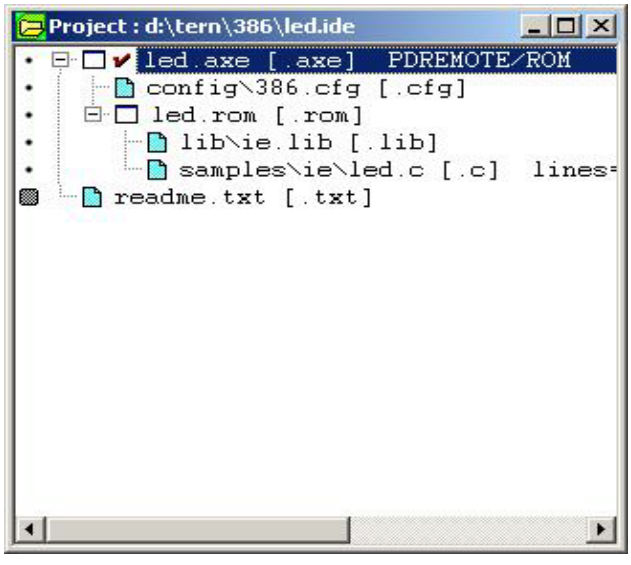
1.5. Open the **Paradigm C/C++** software and if there is any existing project → Click on Project → Close project, then Click on Project → Open project → Browse the project: C:\Tern\586\led.ide and click OK. [Note that the directory (C:\ or D :\) will depend on your installation.]



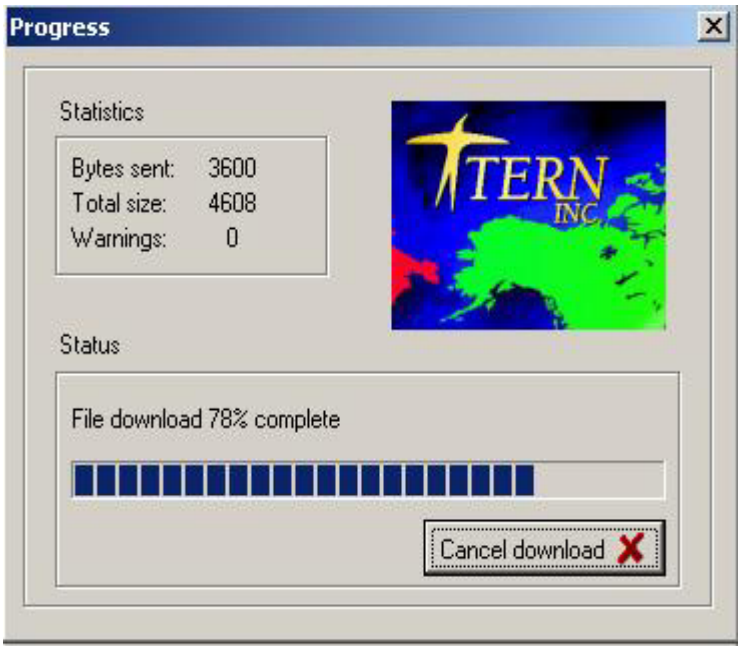
1.6 You can see the following screen:



1.7. Power On the 586 board, Build the project by pressing F9 or Click Project → Make All. Then, Double Click on the led.axe and it will download led.axe to the Tern 586 board.



This is the download page;

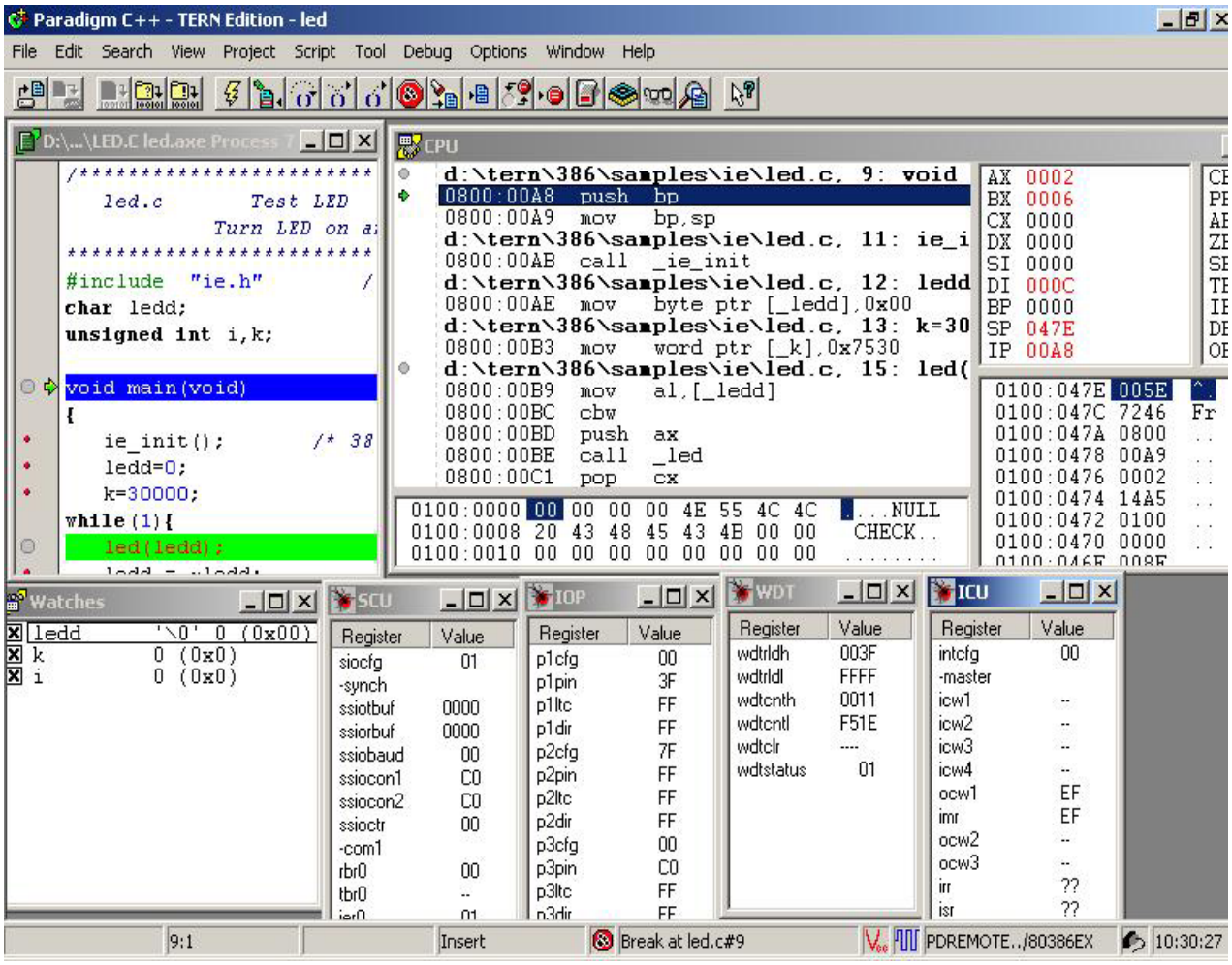


And this is the program,

```
/*********************************************************************

        led.c    Test LED on 586-Engine     10-13-2000

                Turn LED on and off with PIO 0

*********************************************************************/

#include "586.h"        /* 586 initialization */

unsigned char ledd;

unsigned int i,k;

void main(void)

{

        sc_init(); // 586

        ledd=0;

        k=30000;
```

```
while (1){
        led(ledd);  // Directive led(0,1)
        ledd = ~ledd;   // ~ --->> complement
        for(i=0;i<k;i++)
                                {
                                ledd=ledd; // wasting time or delay
                                }
  delay_ms1(10);  // or you can use delay_ms(1000)
        }
}
```

1.8. After Downloading is completed, the Debug Window will appear. And Click on View → then select Target Peripherals you wish to view. Then you can debug as usual like Single stepping, Setting breakpoint



2- Building GPIO project.

2.1 Repeat all the above procedures.

2.2 Copy and past the project led.ide to a LAB1-gio.ide, LAB1-demo, and LAB1-input.

3- Open the lab1-demo.ide project again after closing all the opened projects, then try to delete the program “led.c” and load instate of it “ 586_pio.c”, then save the project

4- Connect J2 with a 40 pins flat IDE cable to the ETS-7000 Digital training kit via the 60 pin adapter.

5- Check the data sheet for J2 connection.

6- The program 586_pio.c is as follow;

```
//*****************************************************************
//      586_pio.c                                       10-13-2000
//              SC520 has 32 Programmable I/O pins: PIO 0-31
//              Each can be programmed as a PIO function or "Interface" function.
//              See SC520 User's Manual(on the CD) Table 23-1 for details
//              sc_init(); initialize after power on, as:
//              P27=/GPCS0=J2.37 for 16-bit I/O operation of on-board ADC/DAC
//              P31=J2.38 as input for STEP2 jumper reading
//              P0 as output for on-board LED control
//              P1=/GPBHE=J1.11 as /BHE for 16-bit data bus high byte enable signal
//              Other 28 PIO pins on the J2 header are free to use:
//              PIO 2-26 and PIO 28,29,30
//
//      Functions for port I/O pins:
// void pio_init(char bit, char mode)
//      where bit=0-31
//                      mode=0, Interface operation
//                      mode=1, Input with pullup/down
//                      mode=2, Output
//
//      void pio_wr(char bit, char dat)
//      where bit=0-31          address the PIO 0-31 pins
//              dat=0/1 controls voltage low/high
//
// unsigned int pio_rd(char port)
//      return status on PIO 15-PIO 0, if port=0
//      return status on PIO 16-PIO 31, if port=1
//
// You can also direct write 16-bit to the PIO register as below:
//      poke(MMCR, _PIOSET15_0_, m); to set PIO 15-0 output pins high
// poke(MMCR, _PIOCLR15_0_, m); to set PIO 15-0 output pins low
```

```
//        poke(MMCR, _PIOSET31_16_, m); to set PIO 31-16 output pins high
// poke(MMCR, _PIOCLR31_16_, m); to set PIO 31-16 output pins low
//*****************************************************************
#include        "586.h"                 /* 586E initialization */

unsigned char i, ledd;
void main(void)
{
 sc_init();         /* 586E initialization */
//        Setup PIO 0=LED, PIO 2-26 and PIO 28,29,30 as output
 pio_init(0, 2); // PIO 0=LED as output
 for(i=2;i<27;i++){
          pio_init(i, 2); // PIO 2-26 as output
          }
 pio_init(28, 2); // PIO 28 as output
 pio_init(29, 2); // PIO 29 as output
 pio_init(30, 2); // PIO 30 as output
 ledd=0;

while(1)
 {
 pio_wr(0,ledd);          // led  on main board PIO 0
 for(i=2;i<27;i++){
          pio_wr(i, ledd); // PIO 2-26 as output
          }
 pio_wr(28,ledd);        // P28
 pio_wr(29,ledd);        // P29
 pio_wr(30,ledd);        // P30
   pio_wr(2,ledd);       // P2---->>> Relay
 ledd=~ledd;
   delay_ms(1000);

 }
     }
```

Compile the project, built, then run the project.

2.4 Check all the pins from P2-P27, P29, and P30 on the training board then on the logic analyzer.

2.5 Update the above C program to generate a 3 second ON then 3 second OFF flashing patterns using P4 till P7 that must connected to board's LEDs or the logic analyzer and oscilloscope. Here repeat the delay statement seven times to get a square wave of **6.120** second period.

2.6 Write a program that generates 20 ms square wave using P10 and show your output on the oscilloscope.

2.7 Connect P10, P11 to LEDs on the training board or your logic analyzer and P12, P13 to the switched, and then develop a program that reads from the input switch and displays on the LEDs. The inputting program as follow:-

```
#include "586.h"          /* 586E initialization */

unsigned  integer a,b;
void main(void)
{
 sc_init ();         /* 586E initialization */

 pio_init(12, 2); // Pin 12 as output
 pio_init(10, 1); // Pin 10  as input
pio_init(13, 2); // Pin 13  as output
pio_init(11, 1); // Pin 11  as input

while (1) {
a = pio_rd(0); // read from 0-15 the lower sixteen
if (a&0x0400) pio_wr(12,1);
else pio_wr(12,0);
if (a&0800) pio_wr(13,1);
else pio_wr(13,0);
}
}
```

5. Questions

Do all the procedures steps very carefully and show all your results to the instructor.

6. Appendixes

1. www.advancedmsinc.com
2. www.tern.com
3. www.testech-elect.com/tern
4. http://www.edaboard.com/forums.html

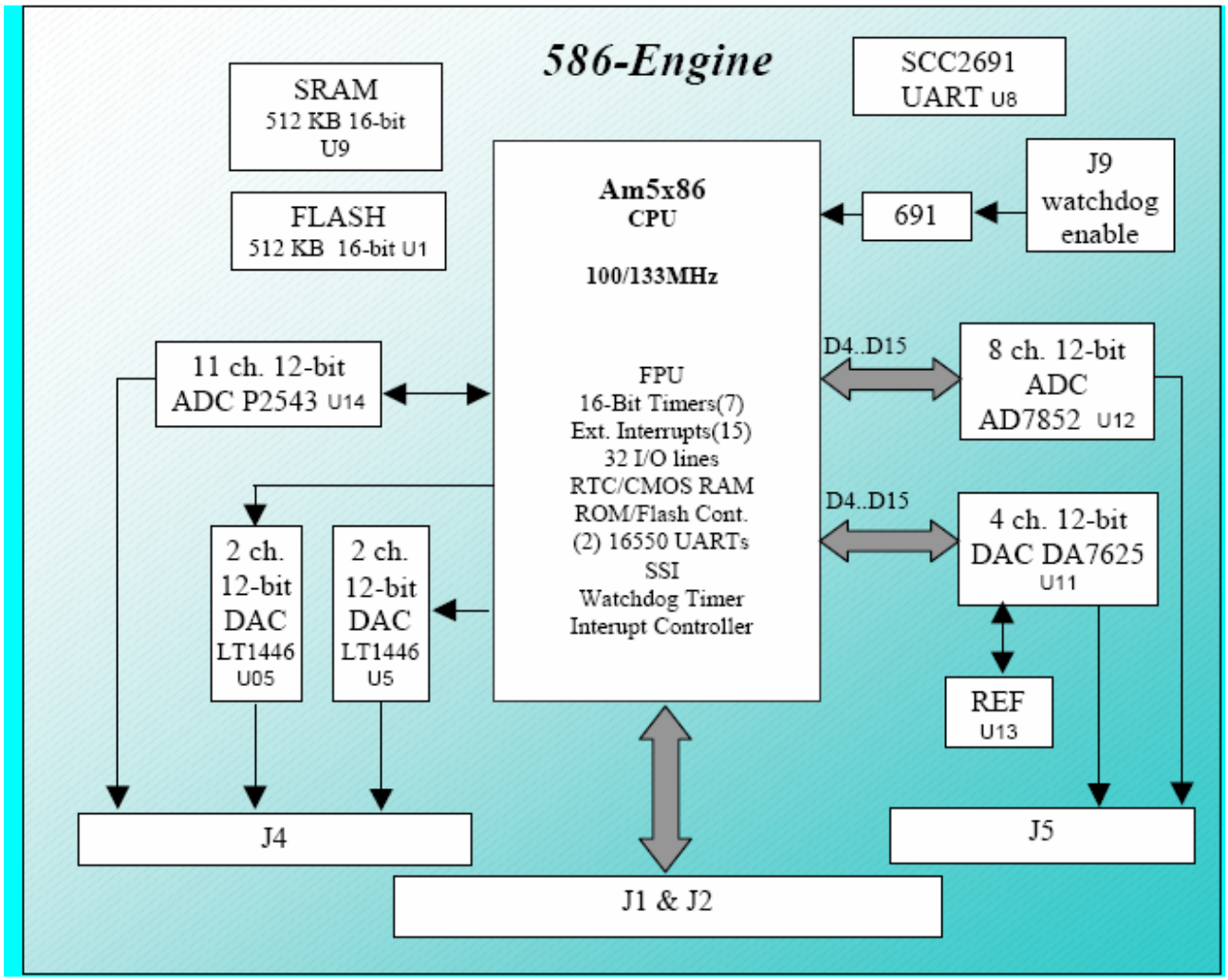
5.

Delay (Counts)	Time (ms)	Frequency (Hz)
25	20	50
50	40	24
100	80	12
200	160	6
300	240	4.2
400	320	3.12
500	400	2.5
600	480	2
800	640	1.5
1000	800	1.25

586 Functional Description

Measuring 3.6 x 2.3 x 0.3 inches, the ***586-Engine(5E)*** is a C/C++ programmable microprocessor module based on a 100/133 MHz, 32-bit CPU (ElanSC520, AMD). Features such as its low cost, compact size, surface-mount flash, high performance floating point coprocessor, and reliability make the ***5E*** ideal for industrial process control and applications requiring intensive mathematical computation. It is designed for embedded applications that require compactness and high reliability. The 586-Engine (5E) integrates an Am586 CPU and high performance ANSI/IEEE 754 compliant hardware floating point unit (FPU). It provides arithmetic instructions to handle various numeric data types and formats and transcendental functions for sine, cosine, tangent, logarithms, etc, useful for intensive computational applications.

Special Note: The core of the Am520 CPU operates at +2.5V and the I/O operation at +3.3V. Also, the input for the I/O is +5V compatible. Stresses above these can cause permanent damage to the SC520 CPU. Operation above these values is not recommended, and can effect device reliability.



Functional block diagram of the 586-Engine

The 586-Engine boots up from on-board 512KB ACTF Flash, and supports up to 512KB battery-backed SRAM. No SDRAM, PCI, or DMA supported. The on-board Flash has a protected boot

loader and can be easily programmed in the field via serial link. Users can download a kernel into the Flash for remote debugging. With the DV and ACTF Flash Kit support, user application codes can be easily field programmed into and run out of the Flash.
A real-time clock* (RTC72423) provides information on the year, month, date, hour, minute, and second, in addition to a 100-year calender and 114 bytes of general purpose battery-backed RAM. This RAM is used by the real-time clock, as well as the ACTF to store the jump address as the pointer to the users application code.
Two industy-standard UARTs support high-speed, reliable serial communication at a rate of up to 1.152 M baud via RS-232 drivers. One synchronous serial interface (SSI) supports full-duplex bi-directional communication. An optional UART SCC2691 may be added in order to have a third UART on-board. All three serial ports support 8-bit and 9-bit communication.
There is one programmable interval timer (PIT) providing 3 16-bit PIT timers and 3 16-bit GP timers, and a software timer. The timers support timing or counting external events. The software timer provides a very efficient hardware timebase with microsecond resolution. In addition, there are two supervisor chips that monitor for power failure, watchdog and system reset.
The 586-Engine provides 32 user-programmable, multifunctional I/O pins from the CPU. Most of the PIO lines share pins with other functions. The 586E supports up to 15 external interrupts. No repair support is available for the 388 pin BGA SC520.
A high-speed, up to 300K samples per second, 8-channel, 12-bit parallel ADC* (AD7852) can be installed. This ADC includes sample-and-hold and precision internal reference, and has an input range of 0-5 V. The 586-Engine also supports a 4-channel, high-speed parallel DAC* (DA7625, 0-2.5V).
An optional 12-bit serial ADC (P2543) has 11 channels of analog inputs with sample-and-hold and a 5V reference that facilitate ratiometric conversion, scaling, and isolation of analog circuitry from logic and supply noise, supporting conversion up to a sample rate of approximately 10 KHz. Up to two optional 2-channel, 12-bit serial DAC (LT1446) that provides 0-4.095V analog voltage outputs capable of sinking or sourcing 5mA is also available. Overall the ***5E*** can support up to 8 analog outputs and 19 analog inputs.

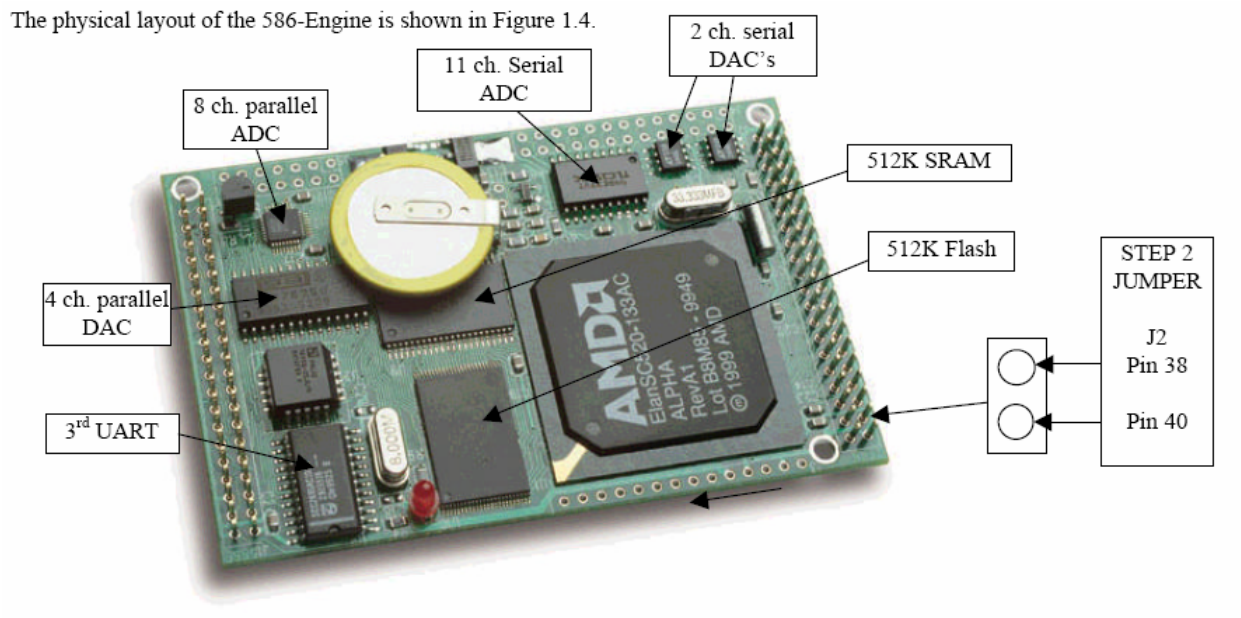
1.2 Features

- Dimensions: 3.6 x 2.3 x 0.3 inches
- Temperature:-40°C to +80°C
- 133MHz, 32-bit CPU (ElanSC520, AMD), Intel 80x86 compatible
- Easy to program in C/C++
- Power consumption: 440 mA at 5V
- Power input: +5V regulated DC, or + 9V to +12V unregulated DC with P100 expansion board installed*
- 512 KB SRAM, 512 KB, 114 byte internal RAM
- 8-channel 300 KHz parallel 12-bit ADC (AD7852) with 0-5V analog input*
- 4-channel 200 KHz parallel 12-bit DAC (DA7625) with 0-2.5V analog output*
- 2 channels serial 12-bit DAC (LT1446), 10 KHz *
- 11 channels serial 12-bit ADC (P2543), 10 KHz *
- high performance floating point coprocessor
- Up to 1GB memory expansion via **MemCard-A**™
- Up to 3 serial ports (2 from ElanSC520, plus one optional SCC2691 UART) support 8-bit or 9-bit asynchronous communication *

- 15 external interrupts with programmable priority
- 32 multifunctional I/O lines from ElanSC520, 1 SSI, 7 16-bit timers
- 114 bytes internal battery-backed RAM. Supervisor (691) for power failure, reset and watchdog
- Real-time clock (RTC72423), lithium coin battery*
- P100 I/O expansion board for regulated 5V power, RS-232/485 drivers, and TTL I/O lines optional

Physical Description

The physical layout of the 586-Engine is shown below



Physical layout of the 586-Engine

SC520 PIOs

The SC520 supports 32 user-programmable I/O lines (PIO). Each of these pins can be used as a user programmable input or output signal, if the interface function is not needed.

The 586-Engine PIO pins are 3.3V output and all inputs are 5V tolerant. Absolutely no voltage greater than 5V should be applied to any pins. Over 5V voltage input can cause permanent damage.

After power-on/reset, PIO pins default to various configurations. The initialization routine, sc_init();, provided by TERN libraries reconfigures some of these pins as needed as:

P27=/GPCS0=J2.37 for 16-bit I/O operation of on-board ADC/DAC

P31=J2.38 as input for STEP2 jumper reading

P0 as output for on-board LED control

P1=/GPBHE=J1.11 as /BHE for 16-bit data bus high byte enable signal

Other 28 PIO pins on the J2 header are free to use. PIO 2-26 and PIO 28, 29, 30 A PIO line can be configured to operate as a output or an input with a weak internal pull-up or pull-down resistor. A PIO pin's behavior, either pull-up or pull-down, is pre-determined and shown in the table below.

These configurations, as well as the processor-internal peripheral usage configurations, are listed below in next Table.

PIO	*Function*	*Power-On/Reset status*	*586-Engine Pin No.*	*586-Engine Initial*
P0	GPALE	Input with pull-up	LED L1 pin 2	Output for LED control
P1	/GPBHE	Input with pull-up	J1 pin 11	High byte enable /BHE
P2	GPRDY	Input with pull-up	J2 pin 4	Input with pull-up
P3	GPAEN	Input with pull-up	J2 pin 1	Input with pull-up
P4	GPTC	Input with pull-up	J2 pin 3	Input with pull-up
P5	GPDRQ3	Input with pull-down	J2 pin 5	Input with pull-down
P6	GPDRQ2	Input with pull-down	J2 pin 7	Input with pull-down
P7	GPDRQ1	Input with pull-down	J2 pin 10	Input with pull-down
P8	GPDRQ0	Input with pull-down	J2 pin 9	Input with pull-down
P9	/GPDACK3	Input with pull-up	J2 pin 12	Input with pull-up
P10	/GPDACK2	Input with pull-up	J2 pin 11	Input with pull-up
P11	/GPDACK1	Input with pull-up	J2 pin 14	Input with pull-up
P12	/GPDACK0	Input with pull-up	J2 pin 13	Input with pull-up
P13	GPIRQ10	Input with pull-up	J2 pin 8	Input with pull-up
P14	GPIRQ9	Input with pull-up	J2 pin 6	Input with pull-up
P15	GPIRQ8	Input with pull-up	J2 pin 16	Input with pull-up
P16	GPIRQ7	Input with pull-up	J1 pin 15	Input with pull-up
P17	GPIRQ6	Input with pull-up	J2 pin 18	Input with pull-up
P18	GPIRQ5	Input with pull-up	J2 pin 17	Input with pull-up
P19	GPIRQ4	Input with pull-up	J2 pin 20	Input with pull-up
P20	GPIRQ3	Input with pull-up	J2 pin 19	Input with pull-up
P21	GPIRQ2	Input with pull-up	J2 pin 21	Input with pull-up
P22	GPIRQ1	Input with pull-up	J2 pin 23	Input with pull-up
P23	GPIRQ0	Input with pull-up	J2 pin 33	Input with pull-up
P24	/GPBUFOE	Input with pull-up	J2 pin 24	Input with pull-up
P25	/GPIOCS16	Input with pull-up	J2 pin 25	Input with pull-up
P26	/GPMCS16	Input with pull-up	J2 pin 29	Input with pull-up*
P27	/GPCS0	Input with pull-up	J2 pin 37	16-bit I/O operation
P28	/CTS2	Input with pull-up	J2 pin 36	Input with pull-up
P29	/DSR2	Input with pull-up	J2 pin 35	Input with pull-up
P30	/DCD2	Input with pull-up	J2 pin 30	Input with pull-up
P31	/RIN2	Input with pull-up	J2 pin 38	STEP2 Jumper

I/O pin default configuration after power-on or reset

C function in the library **586.lib** can be used to initialize and to operate PIO pins.
void ***pio_init***(char bit, char mode);
Where bit = 0-31 and mode = 0 (for interface function), 1 (for input), or 2 (for output).
Example: ***pio_init***(31, 2); will set P31 as output
pio_init(1, 0); will set P1 as /GPBHE

void ***pio_wr***(char bit, char dat);
pio_wr(31,1); set P31 pin high and the LED is off, if P31 is in output mode
pio_wr(31,0); set P31 pin low and the LED is on, if P31 is in output mode
unsigned int ***pio_rd***(char port);
pio_rd (0); return 16-bit status of P0-P15, if corresponding pin is in input mode,
pio_rd (1); return 16-bit status of P16-P31, if corresponding pin is in input mode,
Some of the I/O lines are used by the 586-Engine system for on-board components (Table 3.2). We suggest that you not use these lines unless you are sure that you are not interfering with the operation of
such components (i.e., if the component is not installed).

Signal	Pin	Function
P0	Output	LED control
P1	/GPBHE	High byte D15-D8 data enable of the 16-bit data bus
P27	/GPCS0	General purpose chip select for 16-bit I/O operation
P31	Input	STEP2 jumper

3.5.2 J4 Connector for ADC, DAC

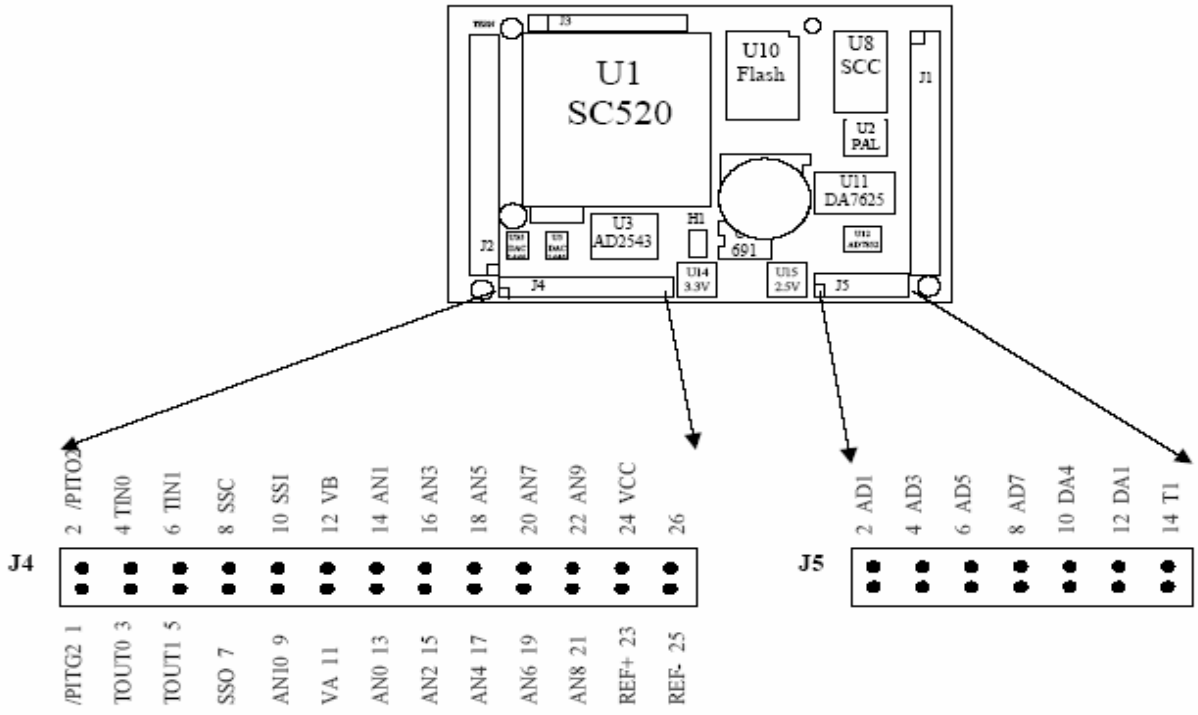


Figure 3.4 J4 connector

Name	Size	Function	Possible Configuration
J1	20x2	Data bus, control bus	MMA, MotionC
J2	20x2	PIO, UART	Pins 38=40: Step2 jumper
J4	13x2	ADC, DAC, SC520 Timers	
J5	7x4	ADC, DAC	

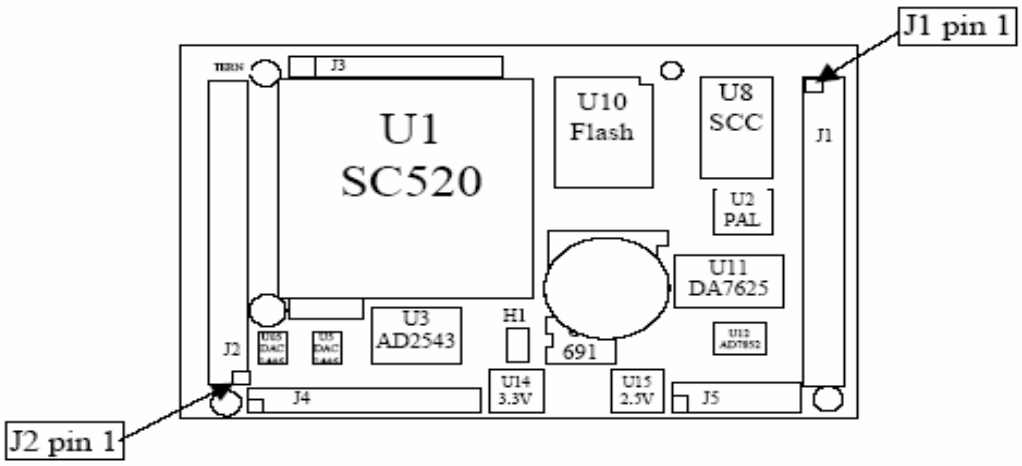


Figure 3.3 Pin 1 locations for J2 and J1

J1 signal

Name	Pin #	Pin #	Name
VCC	1	2	GND
MPO	3	4	CLKT
RxD	5	6	GND
TxD	7	8	D0
VOFF	9	10	D1
/BHE	11	12	D2
D15	13	14	D3
/RST	15	16	D4
RST	17	18	D5
/CS1	19	20	D6
D14	21	22	D7
D13	23	24	GND
	25	26	A7
D12	27	28	A6
/WR	29	30	A5
/RD	31	32	A4
D11	33	34	A3
D10	35	36	A2
D9	37	38	A1
D8	39	40	A0

J2 Signal

Name	Pin #	Pin #	Name
GND	40	39	VCC
P31	38	37	P27
P28	36	35	P29
TxD0	34	33	P23
RxD0	32	31	/RTS1
P30	30	29	P26
TxD1	28	27	/RTS0
RxD1	26	25	P25
P24	24	23	P22
/CTS1	22	21	P21
P19	20	19	P20
P17	18	17	P18
P15	16	15	P16
P11	14	13	P12
P9	12	11	P10
P7	10	9	P8
P13	8	7	P6
P14	6	5	P5
P2	4	3	P4
GND	2	1	P3

J3 signal

Name	Pin #	Pin #	Name
VCC	1	2	GND
V33	3	4	V25
GND	5	6	/DTR2
RIN1	7	8	/DSR1
/DTR1	9	10	/DCD1
/INTA	11	12	/INTB
/INTC	13	14	/INTD

J4 signal

Name	Pin #	Pin #	Name
/PITG2	1	2	/PITO2
TOUT0	3	4	TIN0
TOUT1	5	6	TIN1
SSO	7	8	SSC
AN10	9	10	SSI
VA	11	12	VB